Programación 4

EXAMEN JULIO 2019

Por favor siga las siguientes indicaciones:

- Escriba con lápiz y de un solo lado de las hojas.
- Escriba su nombre y número de documento en todas las hojas que entregue.
- Numere las hojas e indique el total de hojas en la primera de ellas.
- Recuerde entregar su número de examen junto al examen.
- Está prohibido el uso de computadoras, tabletas o teléfonos durante el parcial.

Problema 1 (35 puntos)

Imagine que la Corte Electoral se encuentra desarrollando un nuevo software con el que pretende realizar la gestión de los próximos actos electorales. En la etapa actual del desarrollo se está modelando la componente del sistema que refiere al ingreso de los lemas (sinónimo de partido político) y sus listas.

De cada lema que se desee registrar, interesa conocer un nombre que lo identificará (por ejemplo, Partido de la Ingeniería) y un texto que describe la fórmula presidencial que presenta. Dentro de un lema pueden presentarse diferentes sublemas (por ejemplo, Informáticos Unidos) que serán considerados a la hora de calcular los cargos que se obtienen a nivel parlamentario. De los sublemas se desea conocer el lema al que pertenecen y los mismos datos básicos que posee el lema.

Para cada lema o sublema se pueden registrar listas, que contendrán un número que las identificará y opcionalmente una imagen. Cada lista tendrá un conjunto de integrantes de los cuales interesa saber su nombre completo, sexo, credencial cívica (que los identifica) y posición en la lista (número correlativo).

Adicionalmente, se tiene la descripción del siguiente Casos de Uso:

Caso de Uso:	Ingresar Lista
Actor:	Administrador
Sinopsis:	El caso de uso comienza cuando el Administrador desea ingresar una nueva lista al sistema. Para ello inicialmente el sistema listará los lemas y sublemas para que el administrador seleccione uno. Luego, el Administrador procederá a ingresar el número de lista y la imagen si corresponde. En este momento registrará cada integrante de la lista, en el orden en el que figurará en la misma. Para ello el Administrador ingresa la credencial del integrante. Si ya se ha registrado al sistema un integrante con esa credencial se lo agregará a la lista que se está dando de alta; en otro caso se debe proceder al registro a través del CU Registrar Integrante. Finalmente, deberá confirmar o no el ingreso de la nueva lista.

Se pide:

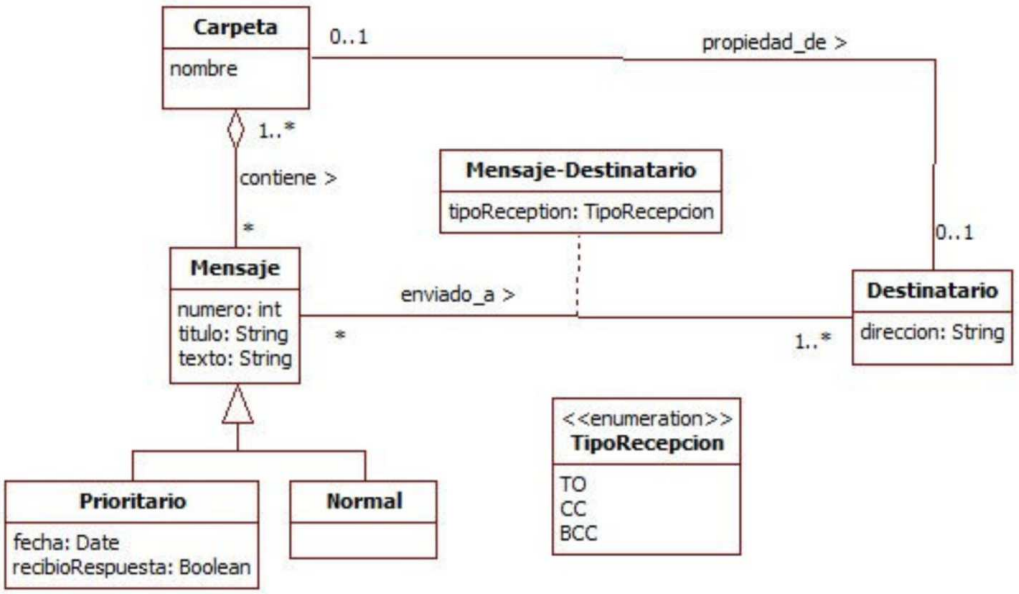
a) Realizar el Modelo de Dominio de la realidad planteada en la descripción de la realidad y el caso de uso. Incluir, si corresponde, restricciones en lenguaje natural.

b) Realizar el Diagrama de Secuencia del Sistema (DSS) para el Caso de Uso Ingresar Lista, indicando el uso de Memoria del Sistema así como de Datatypes.

c) Escribir las pre y post condiciones de los contratos de todas las Operaciones del Sistema del DSS realizado en la parte b).

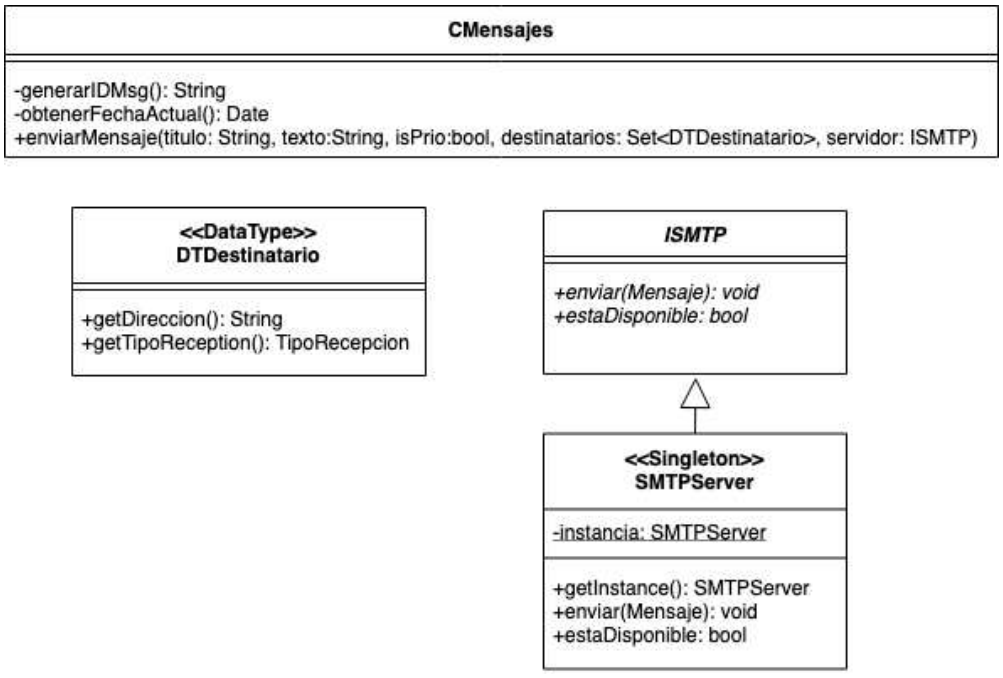
Problema 2 (30 puntos)

Considere el siguiente modelo de dominio y diseño parcial para un sistema de envío de mensajes.



Restricciones:

- El atributo `nombre` identifica a la `Carpeta`.
- El atributo `direccion` identifica al `Destinatario`.
- El atributo `numero` identifica al `Mensaje`.
- La darpeta asociada a un destinatario está compuesta por mensajes enviados a dicho destinatario.
- Existe una `Carpeta` de nombre `ENVIADOS` que contiene todas las instancias de `Mensaje` enviados.



a) **Se pide** realizar el Diagrma de Comunicación correspondiente a la siguiente operación del sistema, incluyendo visibilidades.

`enviarMensaje(titulo: String, texto: String, isPrio: Boolean, destinatarios: Set<DTDestinatario>, servidor: ISMTP)`	
Descripción	Crea un nuevo mensaje en el sistema con el título, texto y prioridad especificados y lo envía a sus destinatarios según el tipo de envío especificado. El envío del mensaje creado es realizado mediante una implementación de ISMTP luego de verificar su disponibilidad mediante la operación `estaDisponible():bool` (true indica que está disponible).
Parámetros	`titulo`: titulo del mensaje. `texto`: contenido del mensaje. `isPrio`: true indica que es un mensaje prioritario, false normal. `destinatarios`: colección de `DTDestinatario` que indica los destinatarios del mensaje y su tipo de recepción correspondiente. `SMTPServer`: servidor de envío de mensajes.
Precondiciones	Existe un `Destinatario` para cada uno de los elementos de destinatarios.
Postcondiciones	Si `isPrio=false`, entonces se crea una instancia de `Normal` con `titulo`, `texto` y número generado por el sistema. Si `isPrio=true`, entonces se crea una instancia de `Prioritario` con `titulo`, `texto`, número y fecha generados por el sistema. Para cada elemento de `destinatarios` se crea una instancia de `Mensaje-Destinatario` con su correspondiente `tipoReception` y se asocia con la instancia de `Mensaje` creado y la de `Destinatario` correspondiente. Se crea un link entre la instancia de `Mensaje` creada y la de `Carpeta` con `nombre="ENVIADOS"`.

b) Se quiere evitar la eventual indisponiblidad de la clase `SMTPServer` para el envío de mensajes, sin modificar su operación `enviarMensaje` y de manera que siempre que verifique disponibilidad se obtenga `true`.

Se pide diseñar una solución (Diagrama de Clases de Diseño, DCD) para la operación `enviarMensaje` basada en el patrón de diseño que considere más adecuado, en la que:

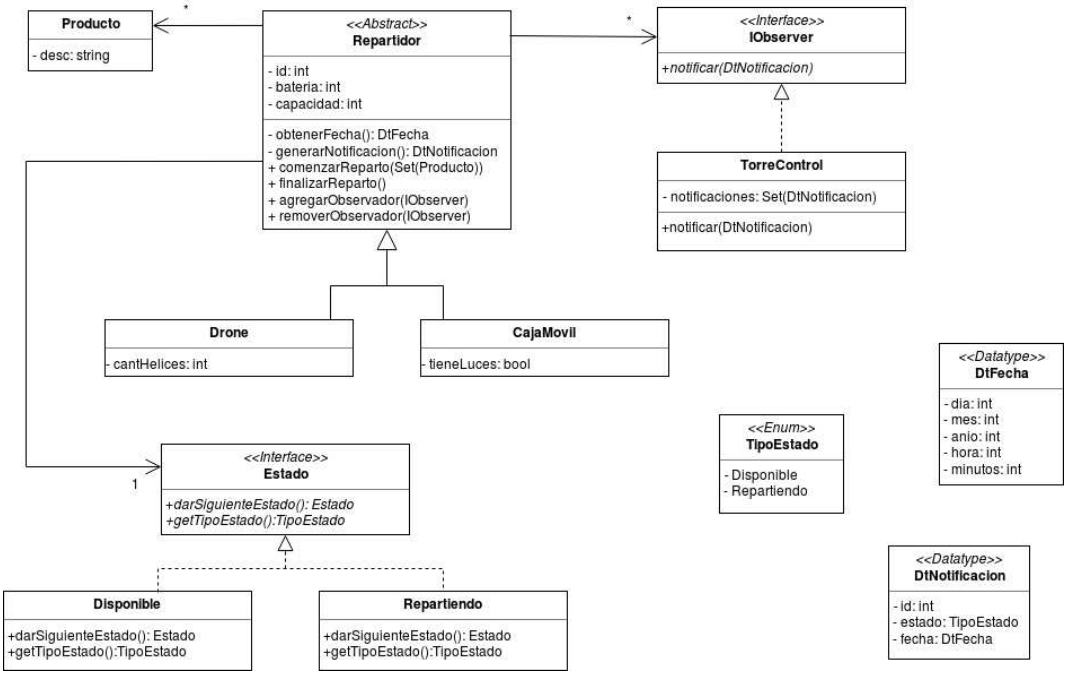
- El envío de mensajes debe realizarse mediante `SMTPServer` y la clase no puede modificarse.
- La clase `CMensajes` no puede ser Singleton ni tener atributos o clases asociadas. Solo se admite la modificación del método `enviarMensaje`.
- Se admite una espera en la entrega de los mensajes enviados mientras los anteriores son procesados, respetando el orden cronológico de envío (FIFO).

Notas:

- Incluya las clases que son parte de la solución de la parte b) únicamente.
- Incorporar en el DCD sólo los constructores o destructores que se utilicen en los diagramas de comunicación. No incorporar al DCD las operaciones de colecciones.

Problema 3 (35 puntos)

Se desea construir un sistema para manejar repartidores autónomos (por ejemplo, drones) de productos a personas. Para dicha tarea, el equipo de desarrollo llegó al siguiente Diagrama de Clases de Diseño (se omiten las dependencias).



El comportamiento de las operaciones de cada clase se describe a continuación:

`TorreControl`:
- `notificar(DtNotificacion)`: Guarda una nueva notificación para su posterior uso.

`Repartidor`:
- `comenzarReparto(Set(Producto))`: Agrega los productos al repartidor, cambia el estado al correspondiente y notifica a los observadores. El id de la notificación debe ser el mismo que el del repartidor que la emite.
- `finalizarReparto()`: Elimina los productos del repartidor, cambia el estado al correspondiente y notifica a los observadores. El id de la notificación debe ser el mismo que el del repartidor que la emite.
- `agregarObservador(IObserver)`: Agrega un nuevo observador al cual notificar.
- `removerObservador(IObserver)`: Remueve un observador.
- `generarNotificacion()`: Retorna un datavalue de tipo DtNotificacion con los datos actuales del repartidor.

`Disponible`:
- `darSiguienteEstado()`: Devuelve una instancia del siguiente estado.
- `getTipoEstado()`: Retorna un valor enumerado correspondiente al estado.

Se pide:
- Implementar en C++ las interfaces `IObserver` y `Estado` e implementar completamente (.h y .cpp) las clases `TorreControl`, `Repartidor`, `Drone` y `Disponible`.

Considerar:

- La operación `obtenerFecha` ya se encuentra implementada para la clase `Repartidor`.
- Es posible utilizar las clases `set<T>`, `map<K,V>` y `vector<T>` de `STL`.
- Puede suponer la existencia de la interface `ICollectible` e implementaciones de `ICollection` (clase `List`) e `IIterator` según sea necesario.
- Las implementaciones deben incluir constructores y destructores, liberando en estos últimos toda la memoria que sea necesaria.
- Asumir existencia y no implementar las operaciones get y set de los atributos.
- No incluir directivas al precompilador.
- No implementar los datatypes.